ARMY RESEARCH LABORATORY

RVis: The RIVA/MUVES Prototype Visualization Tool

by Jason L. Owens and Lee A. Butler

ARL-TN-223 **August 2004**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5068

ARL-TN-223 **August 2004**

RVis: The RIVA/MUVES Prototype Visualization Tool

Jason L. Owens and Lee A. Butler
Survivability/Lethality Analysis Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
August 2004	Final	August 2003–December 2003

4. TITLE AND SUBTITLE
RVis: The RIVA/MUVES Prototype Visualization Tool

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Jason L. Owens and Lee A. Butler

5d. PROJECT NUMBER
1L162618AH80

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: AMSRD-ARL-SL-BE
Aberdeen Proving Ground, MD 21005-5068

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TN-223

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

RVis was created to prototype a visualization tool for the vulnerability analyst community and to provide a starting point for discussion on the uses and design of such a tool in the Real-Time Interactive Vulnerability Analyzer (RIVA)/Modular UNIX-based Vulnerability Estimation Suite (MUVES). This document describes the initial requirements, design, and technical aspects of the version that was demonstrated to a segment of the community in December 2003.

15. SUBJECT TERMS
visualization, MUVES, 3-D modeling, intermediate results, Cocoa

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Jason L. Owens
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL	22	19b. TELEPHONE NUMBER (*Include area code*) 410-278-6736

Contents

List of Figures

List of Tables

1. Project Background

In August 2003, members of the Advanced Computer Systems Team of the U.S. Army Research Laboratory's Survivability/Lethality Analysis Directorate (SLAD) were tasked to continue earlier work to better visualize results from the Modular UNIX*-based Vulnerability Estimation Suite (MUVES). This work was produced using the Tool Command Language and the Visualization ToolKit (VTK). The objective for this effort was to develop a useful prototype in a rapid, iterative, and incremental manner. In addition, after prototyping was complete, a dialogue with the target user community and further visualization efforts could be initiated.

1.1 Motive for Work

The current state-of-the-art for visualizing MUVES results files is two-dimensional (2-D) cell plots. The cell plots represent color-coded values from a specific state vector,† from binary values of hit or no-hit, to real-valued probabilities of kill. A cell plot directly corresponds to the results of a view and its corresponding aim points and firing points. Thus, a view at 0° azimuth, 0° elevation would be looking straight on at the front of most models (see figure 1), and the cell plot for a full analysis would look like the profile at that orientation. On the other hand, a side-on view (90º, 0º) would produce a larger profile.

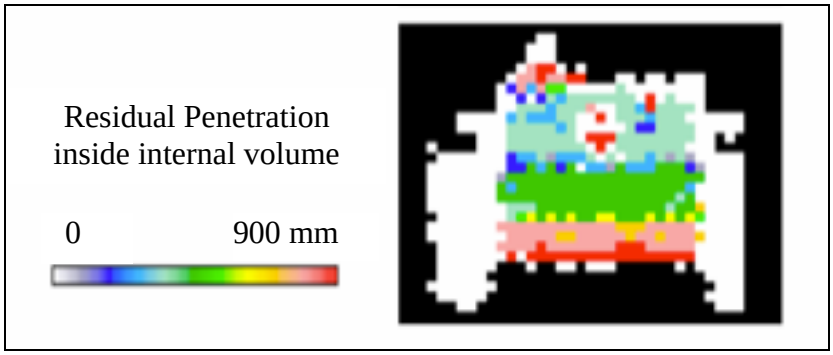

Figure 1. A sample cell plot.

Unfortunately, the current analysis runs in three dimensions (3-D) and operates on highly detailed computer models of a target, whereas the visualization tools are in 2-D. This means that the current tools (1) immediately throw away a large portion of the data and (2) reduce the efficiency in which one can recognize and process information. Thus, a natural question presents itself: "What can we show in 3-D?"

* UNIX is a registered trademark of The Open Group.

† A state vector contains a mapping of MUVES components to values for a particular aspect of interest (e.g., hit, probability of kill, or loss of function for a particular firing point and its resulting trace path). Analyses typically contain multiple state vectors containing specific "critical" components.

1.2 Choice of Development Environment

To support the "rapid" portion of the previously stated project objective, Apple's advanced object-oriented (OO) framework, Cocoa, was chosen as the development environment for the work (see http://developer.apple.com/cocoa [Apple Computer[1]]). Although other platforms were initially considered for the task, there were several reasons for this choice. First, Cocoa is arguably the fastest way to build powerful graphics-based applications. It is mature and well-supported (having been born as NeXTSTEP), and it runs on the U.S. Army Research Laboratory's (ARL's) fastest desktops, the PowerMac dual G5s. In addition, although Cocoa is not platform-independent and uses a not-so-widely-employed (albeit powerful) language called Objective-C, these limitations could also be considered strengths, depending on one's point of view. Finally, porting a well-designed application from one environment to another is considered to be a small task relative to the original coding time, and refactoring while porting is a useful technique for improving the code design efficiently.

Nonetheless, the authors believe that the next development environment should be totally cross-platform and provide sufficient capability to develop user interfaces. Unfortunately, there are not many environments that currently fall into this category. Trolltech's QT and Sun's Java are the closest matches, and each has its own pros and cons (see http://www.trolltech.com /products/qt/index.html [Trolltech AS[2]] and http://java.sun.com [Sun Microsystems[3]]).

2. Initial Requirements

Because the project began with the high-level goal of rapidly developing a usable prototype, requirements were discussed only until both parties agreed on the initial features. Each "iteration" of development was used to further clarify requirements, discuss new features, and address any issues arising during development. Although this type of approach results in relatively informal requirements, it also results in a product that better represents actual desired features (as compared to approaches with a more formal requirements process). In addition, it helps avoid the all-too-typical situation in which a user community has difficulty imagining and identifying specific requirements and features (e.g., in the graphical user interface) for a tool that does not yet exist.

Table 1 summarizes the functional and nonfunctional factors that ultimately drove RVis design and development decisions.

[1] Apple Computer. Cocoa. http://developer.apple.com/cocoa (accessed 15 April 2004).

[2] Trolltech AS. Qt Overview. http://www.trolltech.com /products/qt/index.html (accessed 15 April 2004).

[3] Sun Microsystems. Java Technology. http://java.sun.com (accessed 15 April 2004).

Table 1. Functional and nonfunctional requirements.

Requirement Type	Requirement No.	Requirement
Functional	1a	Render target in shaded real-time 3-D view.
	1b	Process and display MUVES intermediate and final results files (since each file provides information the other does not).
	1c	Display state vectors for selection.
	1d	Display view data (firing points) in 2-D and 3-D.
	1e	Display the shotline traces in the 3-D view.
	1f	Display color-coded component interactions for a particular shotline and state vector selection (e.g., "Critical Component PKs").
	1g	Provide view manipulation capabilities, including camera orientation and component visibility/translucency.
Nonfunctional	2a	Make all manipulations as real-time as possible and provide feedback when necessary.
	2b	Make all results accessible from within the program itself (i.e., no need to look in the actual files).
	2c	Present a maximum amount of data without overloading the display.
	2d	Make the tool easy to learn and use.

3. Design Description

3.1 Architecture Overview and Programming Tasks

Figure 2 provides an overall picture of the RVis' internal components and collaboration. The code and programming tasks for the project were divided into three basic areas: (1) results file parsing, (2) component display, and (3) actual interface construction (including the OpenGL widget integration). These areas and the enhancement summaries are discussed in the next subsections.

3.1.1 Results File Parsing

Results file parsing was perhaps the most time-consuming development task in the project. Because files can be fairly large (100–500 MB) for average analyses, RVis initially scans and indexes both files simultaneously and stores the index for future use. This means faster application start-up and overall quicker response times while operating the tool. The most difficult part of this task came in understanding the MUVES file formats because documentation was quite minimal and the existing parsing routines were not suitable for this task. In addition to parsing and indexing, RVis processes the data as needed at runtime into an appropriate hierarchical data structure, for both browsing and facilitating shotline analysis.

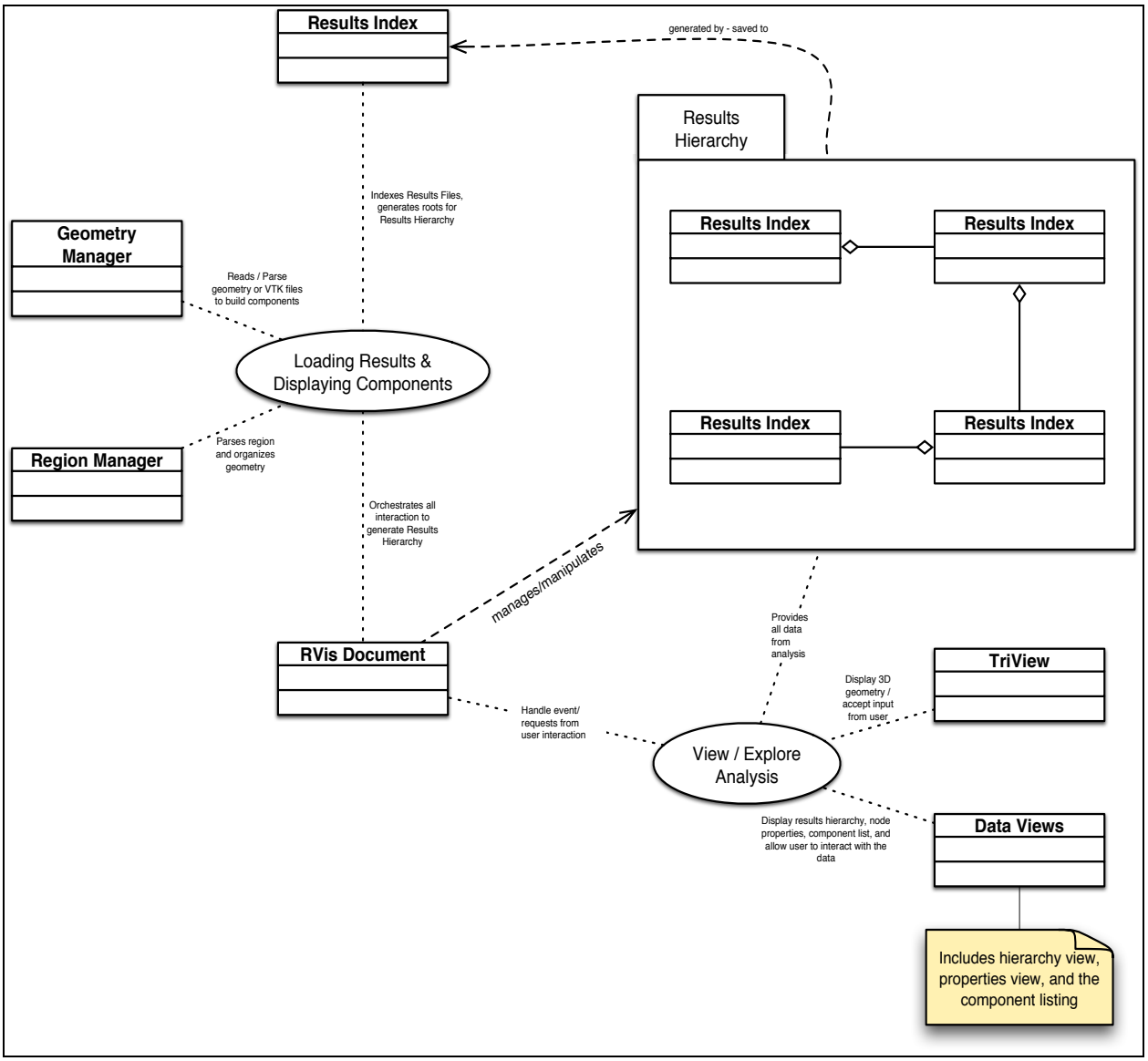

Figure 2. Collaboration overview.

3.1.2 Component Display

Component display was a slightly easier task, and RVis actually supports two kinds of input: (1) native BRL-CAD geometry files and (2) preparsed VTK-format polygon files. In practice, one should rarely use the first type because processing the U.S. Army Ballistic Research Laboratory-Computer-Aided Design (BRL-CAD*) geometry into polygonal surfaces is quite time-consuming (on the order of several hours, depending on complexity). However, two additional command-line tools that perform two separate tasks are included. The first, **`bot_db`**, can output VTK files for each region in a BRL-CAD model. The second, **`vtk_rv`**, can process those individual VTK files (along with an appropriate region_map file) into binary RVis components.

* BRL-CAD is a registered trademark of ARL.

RVis components can be read directly by the tool for quick model loading and contain additional settings and data for use by the tool itself (e.g., object color, shell part, opacity, etc.).

3.1.3 Interface Construction

Although the current OpenGL widget, called TriView, was already partially developed before the project began, it was enhanced to support some of the requirements. The newly included features are trackball, field-of-view (zoom), and pan, as well as other technical capabilities, such as supporting alternate geometries for live window resize and quasimodes (key-press/mouse movement combinations). Other important widgets used throughout the interface are trees and tables, which are requisite for the display of complex hierarchical and detailed data.

3.2 Functional Requirement Satisfaction

The following "graphical" explanation illustrates several views of the RVis tool and provides a brief description of how each of the functional requirements enumerated in table 1 was satisfied.

3.2.1 Full Window Display

Figure 3 illustrates a standard view of the application execution. The main view on the right satisfies the requirement to render a target in shaded real-time 3-D view (requirement 1a in table 1). The two panels on the left of the main window satisfy the requirements to process and display MUVES intermediate and final results files (requirement 1b in table 1) and to display state vectors for selection (requirement 1c in table 1), respectively.

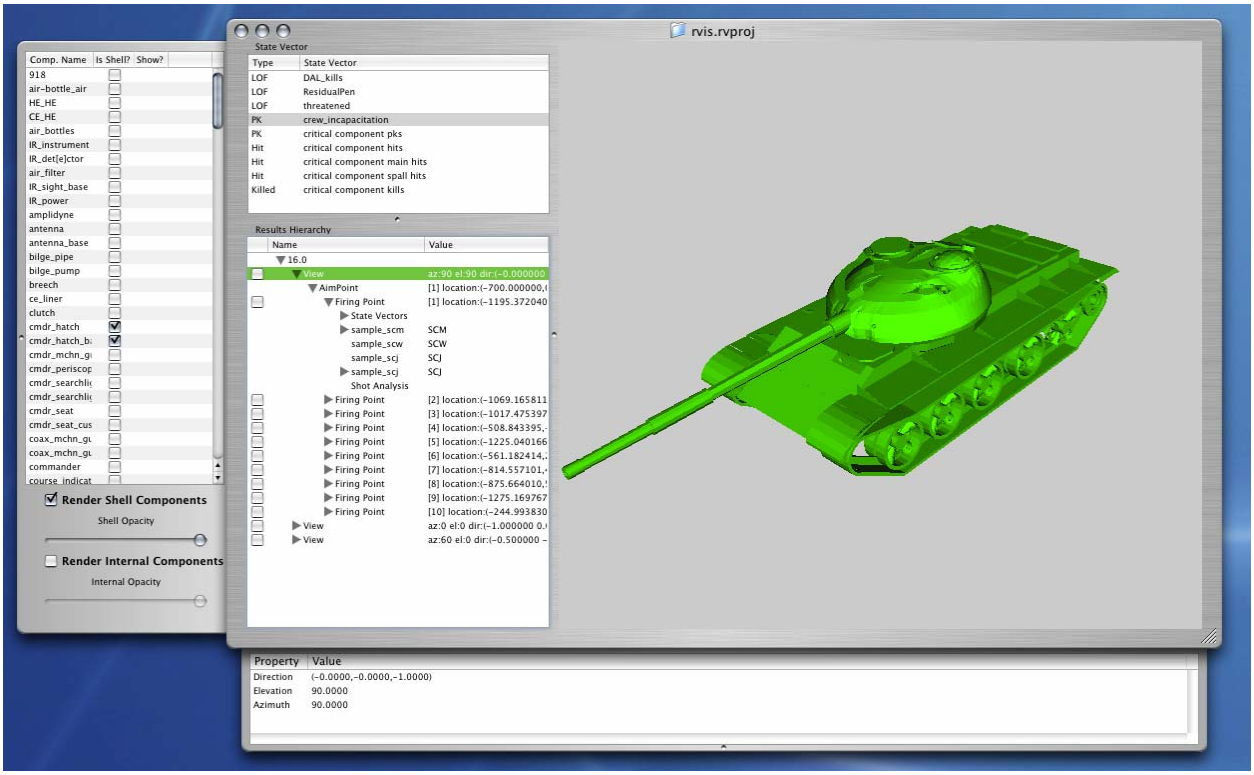

Figure 3. Full window display.

3.2.2 Firing Point Detail

Figure 4 shows a detail of the main window and 2-D firing point display when a view has been selected from the results hierarchy. This capability satisfies the requirement to display view data (firing points) in 2-D and 3-D (requirement 1d in table 1). Note that the floating window in the upper left corner displays a transparent 2-D view of the firing points in proper orientation, while the 3-D view also displays firing point indicators relative to the actual target model. The coloring is based on the well-tested "heated body" color scale. This scale is similar to modified blackbody radiation and increases linearly in saturation, brightness, and hue. This functionality gives the viewer a better chance to distinguish between similar values, especially in a shaded 3-D environment.

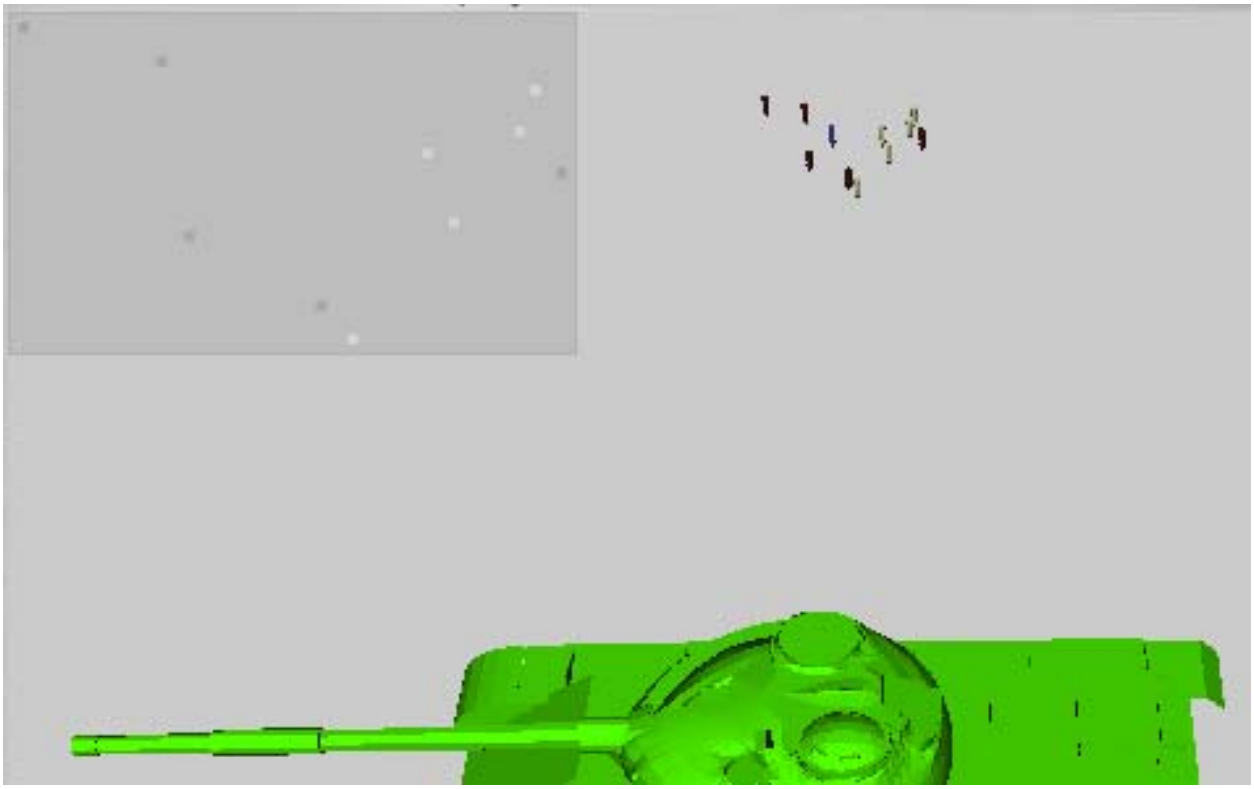

Figure 4. Firing point detail in 2-D and 3-D.

Figure 5 shows another view of the 2-D firing point display. This time, the display has been enlarged and made fully opaque for the purpose of selecting a firing point of interest. Figure 6 shows a close-up of the model display including the shotlines rendered for a particular firing point. Notice that the components interacting with a shot trace are colored according to the value assigned to them in the state vector. This functionality satisfies the requirements to display the shotline traces in the 3-D view (requirement 1e in table 1) and to display the color-coded component interactions for a particular shotline and state vector selection (requirement 1f in table 1).

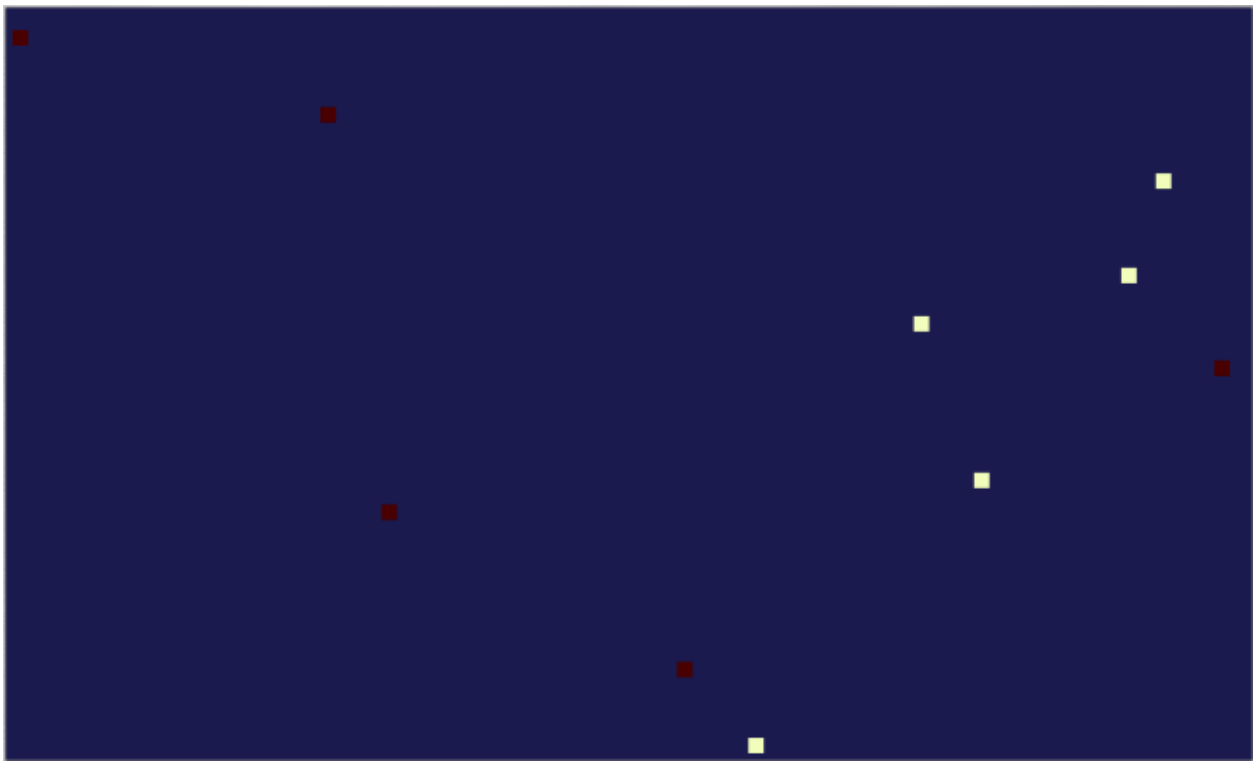

Figure 5. Firing point overlay in 2-D.

3.2.3 TriView Manipulations

Figure 7 shows the satisfaction of the final functional requirement—to provide view manipulation capabilities, including camera orientation and component visibility/translucency (requirement 1g in table 1). The view manipulation modes are presented in a context-sensitive pop-up menu available with the right button. Future versions of RVIs are planned to include the ability to use keypresses to trigger a temporary manipulation mode, while still maintaining a user-selected default mode.

3.3 Nonfunctional Requirement Support

Unlike the functional requirements, the stated nonfunctional requirements are difficult to quantify without at least minimal user testing (for which RVis is not yet ready) and thus cannot be directly satisfied (or illustrated) by a particular feature implementation. However, the following subsections summarize the RVis design/development choices that were made in order to support the intent of the nonfunctional requirements.

3.3.1 Real-Time Manipulations With Feedback

In support of the requirement to make all manipulations as real-time as possible and to provide feedback when necessary (requirement 2a in table 1), users are now enabled to choose their own "shell" representation of a target (i.e., picking only those external elements they would like to see

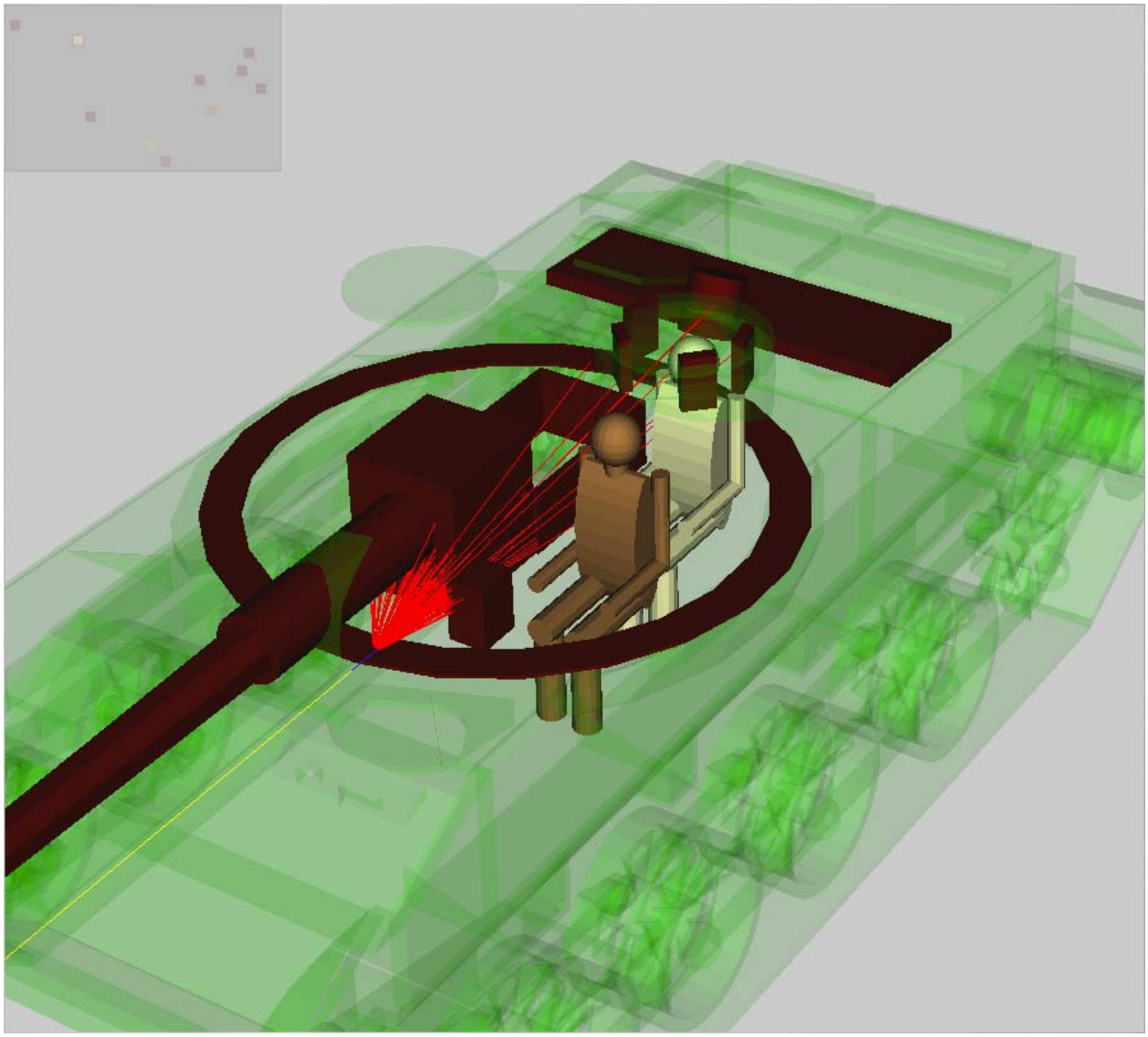

Figure 6. Shotline detail.

to maintain orientation). This capability reduces the overall amount of geometry that is required to be rendered each frame. In addition, "quick object" rendering is provided by the TriView widget, which allows the programmer to specify *simplified* geometry for rendering during a live resize. Some processing is also performed in multiple threads in order to take advantage of multiple processor systems and to prevent the interface from feeling sluggish. (Note that section 5 of this document discusses possibilities for future development of this feature.)

3.3.2 Results Accessibility

The current results hierarchy, in tandem with the properties drawer, textually satisfies the requirement to make all results accessible from within the program itself (requirement 2b in table 1). However, some details are not yet represented in graphical form, but they may not need to be so. More user feedback will be necessary to determine any additions.

Figure 7. Full window display with view options.

3.3.3 Display Density

In support of the requirement to present a maximum amount of data without overloading the display (requirement 2c in table 1), the 3-D view of the target and shotlines already compresses the amount of data one can digest into a manageable space (it's virtually impossible to visualize those relationships in one's head just by reading the files). However, other efforts have been made in the "only display what's needed" approach. The option to display properties only when they are relevant (by use of the Cocoa "drawer" component) minimizes extraneous display; the use of an omnipresent floating 2-D firing point display enables the analyst to quickly select firing points of interest and provides a visualization tool similar to the cell plots.

3.3.4 Ease of Use

In support of the final nonfunctional requirement to make the tool easy to learn and use (requirement 2d in table 1), most key commands use easy-to-remember mnemonics (such as pressing the "f" key to activate the firing point selection window). However, more work may need to be done in this area because building the visualization projects themselves are difficult, and not all options in the interface are readily apparent (e.g., the context-sensitive view menu).

4. Initial Demonstration

In December 2003, the RVis prototype was demonstrated to a small group of vulnerability analysts and other SLAD personnel. The demonstration served primarily to introduce the new visualization concept to those who may not have been aware of its existence. It also proved to be a good starting point for feedback on what other features may be needed to make the tool useful.

Because RVis represented a significant change from the classic business process, it would not have been unusual for demonstration attendees to initially view the prototype with a degree of caution. However, after some initial discussion about certain implementation choices (e.g., the heated body coloring scale vs. the traditional rainbow scale), attendees generally seemed impressed with the tool's features.

Discussions during the meeting centered around the functionality of existing features, requests for additional features to enhance the tool's utility, and concerns about the current state of the analysis process. The next subsections summarize the latter two topics.

4.1 Additional Feature Requests

The following additional tool capabilities were requested during the demonstration.

- Handle averaging multiple firing point iterations, possibly over multiple runs, and provide the capability to toggle between options.
- Provide for some visualization of air interaction. The system currently recognizes air but is not drawing the geometry.
- Toggle spall visibility (in order to view main penetrator without clutter).
- Provide the ability to assign colors to various trace types (e.g., mass/velocity spall fragment [MVSF] or casing fragments).
- Toggle trace contributions to a selected or group of selected components (i.e., show only those traces that contribute to a component's state) and toggle intermediate component interaction visibility.
- Display a list of objects along a selected shotline.
- Generate screenshots for any particular view.
- Generate a movie. More user consultation would be needed to determine exactly how to generate this movie (i.e., would it be just a recording of user actions or a flight along the shotline?).

- Provide a shotline radius setting. In addition, it may be useful to generate a shotline radius based on penetrator/fragment volume.
- Provide color scale legends for any colored data.

4.2 Current Analysis Process Concerns

The following items were proposed during the meeting as being areas of concern about the current analysis process.

- Selecting the generation of intermediate results (IR) files that RVis uses causes a substantial increase in the runtime of the MUVES process.
- IR files are quite large.

Although these concerns are valid and should be considered in any future development work, they may actually stem more from current MUVES configuration and hardware issues than from the RVis implementation itself. For example, with regard to the slow runtime, the authors postulate (but have not confirmed) that a significant amount of overhead is being spent converting the data from binary to text prior to output. Another possibility is that MUVES is not using buffered input/output. Also, with regard to large file sizes, some users' definition of "large" seems to relate more to the use of relatively small disk drives for MUVES runs and the amount of text they are willing to look at in a text editor.

Thus, possible resolutions to these issues include (1) upgrading to more modern disk drives that can better handle today's larger files, (2) using an outboard process to get idle processors (at least in multiprocessor machines) to compress the files and reduce runtime, and (3) producing a binary version of the output file to avoid the overhead of converting the data to ASCII (with appropriate postprocessing tools for effective examination of this data).

5. Future Work

Because RVis is a prototype (albeit a useful one), there is still much work to be done to make it a full-fledged application. Besides the previously mentioned additional features that were requested during the demonstration, there are some fundamental items still missing. The following list highlights some of the areas identified for future enhancement and the estimated task priority.

- **Portability of RVis**. Because RVis was built as a prototype, it may need to be ported to a different language or application programming interface. Any major future development on this tool will naturally take into consideration multiplatform portability as well as interactions with future software, such as the RIVA/MUVES3 platform (*high priority*).

- **"Project Wizard" Capability.** Currently, a project (the term used to describe all the files needed to run a visualization session) must be cobbled together by hand; manually building the directory structure and copying or generating the right files are time consuming and error prone. A script or external program, which would run in the background and perform all necessary tasks, could be created to make this process easier (*low priority*).
- **Threaded Data Accesses to the Results Index**. All data accesses to the results index must be threaded. This should be done to ensure the user does not lose control of the application, especially if there has already been interaction with the view. As analyses get bigger, the index will take longer to load for any given firing point. In addition, loading processes should be evidenced with appropriate feedback (e.g., a spinning beach ball or progress bars) (*medium priority*).
- **More View Manipulation Functions**. "Snap-to" functions are needed to center the view on areas of interest, thereby saving the user time in configuring the view for common elements, such as penetrator entry point, spall origination points, or even framing individual components (*low priority*).
- **Built-In Simple Help or Tutorial Documentation**. The help system could be as straightforward as launching an HTML page or providing embedded tips with a key press over an item of interest (*medium priority*).
- **Enhanced TriView**. TriView (the OpenGL widget) began as a simple extended Cocoa component to quickly display geometry, but it was developed without clear design goals. A reworked version with better design would prove more useful and reusable than the current object. The main goal would be to provide a very thin platform-dependent OpenGL view that relied on a platform-independent, scene-based object model. Not only would it enforce OO paradigms, but it would also promote code portability (*medium priority*).

NO. OF COPIES	ORGANIZATION
1 (PDF Only)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218
1	COMMANDING GENERAL US ARMY MATERIEL CMD AMCRDA TF 5001 EISENHOWER AVE ALEXANDRIA VA 22333-0001
1	INST FOR ADVNCD TCHNLGY THE UNIV OF TEXAS AT AUSTIN 3925 W BRAKER LN STE 400 AUSTIN TX 78759-5316
1	US MILITARY ACADEMY MATH SCI CTR EXCELLENCE MADN MATH THAYER HALL WEST POINT NY 10996-1786
1	DIRECTOR US ARMY RESEARCH LAB AMSRD ARL CS IS R 2800 POWDER MILL RD ADELPHI MD 20783-1197
3	DIRECTOR US ARMY RESEARCH LAB AMSRD ARL CI OK TL 2800 POWDER MILL RD ADELPHI MD 20783-1197
3	DIRECTOR US ARMY RESEARCH LAB AMSRD ARL CS IS T 2800 POWDER MILL RD ADELPHI MD 20783-1197

NO. OF COPIES	ORGANIZATION
	ABERDEEN PROVING GROUND
1	DIR USARL AMSRD ARL CI OK TP (BLDG 4600)

NO. OF COPIES ORGANIZATION

1 OASD C3I
J BUCHHEISTER
RM 3D174
6000 DEFENSE PENTAGON
WASHINGTON DC 20301-6000

1 OUSD(AT)/S&T AIR WARFARE
R MUTZELBURG
RM 3E139
3090 DEFENSE PENTAGON
WASHINGTON DC 20301-3090

1 OUSD(AT)/S&T LAND WARFARE
A VIILU
RM 3B1060
3090 DEFENSE PENTAGON
WASHINGTON DC 20310-3090

1 UNDER SECY OF THE ARMY
DUSA OR
RM 2E660
102 ARMY PENTAGON
WASHINGTON DC 20310-0102

1 ASST SECY ARMY
ACQSTN LOGISTICS & TECH
SAAL ZP RM 2E661
103 ARMY PENTAGON
WASHINGTON DC 20310-0103

1 ASST SECY ARMY
ACQSTN LOGISTICS & TECH
SAAL ZS RM 3E448
103 ARMY PENTAGON
WASHINGTON DC 20310-0103

1 DIRECTOR FORCE DEV
DAPR FDZ
RM 3A522
460 ARMY PENTAGON
WASHINGTON DC 20310-0460

1 US ARMY TRADOC ANL CTR
ATRC W
A KEINTZ
WSMR NM 88002-5502

1 USARL
AMSRD ARL SL M
J PALOMO
WSMR NM 88002-5513

NO. OF COPIES ORGANIZATION

1 USARL
AMSRD ARL SL EA
R FLORES
WSMR NM 88002-5513

1 USARL
AMSRD ARL SL EI
J NOWAK
FT MONMOUTH NJ 07703-5601

ABERDEEN PROVING GROUND

1 US ARMY DEV TEST COM
CSTE DTC TT T
APG MD 21005-5055

1 US ARMY EVALUATION CTR
CSTE AEC SVE
R BOWEN
4120 SUSQUEHANNA AVE
APG MD 21005-3013

1 US ARMY EVALUATION CTR
CSTE AEC SVE S
R POLIMADEI
4120 SUSQUEHANNA AVE
APG MD 21005-3013

1 US ARMY EVALUATION CTR
CSTE AEC SV L
R LAUGHMAN
4120 SUSQUEHANNA AVE
APG MD 21005-3013

13 DIR USARL
AMSRD ARL SL
J BEILFUSS
P DEITZ
AMSRD ARL SL B
J FRANZ
M PERRY
P TANENBAUM
AMSRD ARL SL BB
D BELY
D FARENWALD
S JUARASCIO
M RITONDO
AMSRD ARL SL BD
R GROTE

NO. OF
COPIES ORGANIZATION

AMSRD ARL SL BE
L ROACH
AMSRD ARL SL E
M STARKS
AMSRD ARL SL EC
J FEENEY
E PANUSKA

NO. OF
COPIES ORGANIZATION

1 NAWC
WEAPONS DIV
CODE 418300D A WEARNER
BLDG 91073
1 ADMINISTRATION CIR
CHINA LAKE CA 93555-6100

1 USAF ARMAMENT CTR
AAC/ENMA
D MCCOWN
101 W EGLIN BLVD
EGLIN AFB FL 32542-5549

1 USAF
46 OG OGMLV
B THORN
104 CHEROKEE AVE
EGLIN AFB FL 32542-5600

1 USAF WRIGHT LAB
46TH OG OGM AL AC
M LENTZ
2700 D ST BLDG 22B
WRIGHT PATTERSON AFB OH
45433-7605

ABERDEEN PROVING GROUND

2 DIR USARL
AMSRD ARL SL
C HARDIN
AMSRD ARL SL E
D BAYLOR